Centro Universitário Santo Agostinho

www4.fsanet.com.br/revista
Rev. FSA, Teresina, v. 20, n. 10, art. 9, p. 175-202, out. 2023
ISSN Impresso: 1806-6356 ISSN Eletrônico: 2317-2983
http://dx.doi.org/10.12819/2023.20.10.9

Métodos Heurísticos e Meta-Heurísticos para a Resolução do Problema de Sequenciamento de Ordens de Manutenção Preventiva de Longo Prazo

Heuristic and Meta-Heuristic Methods to Resolve the Long-Term Preventive Maintenance Scheduling Problem

Arthur Almeida Santos
Mestrado em Engenharia de Produção pela Universidade Federal de Ouro Preto
Graduação em Engenharia de Produção pela Universidade Federal de Viçosa
arthur.jf.mg@hotmail.com

Alexandre Xavier Martins
Doutor em Engenharia Elétrica pela Universidade Federal de Minas Gerais
Professor da Universidade Federal de Ouro Preto - UFOP
xmartins@ufop.edu.br

Marcone Jamilson Freitas Souza
Doutor em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro
Professor da Universidade Federal de Ouro Preto - UFOP
MG marcone@ufop.edu.br

Rafaela Heloisa Carvalho Machado
Doutorado em Engenharia de Produção pela Universidade Federal de Minas Gerais
Professora da Universidade Federal de Viçosa - UFV
rafaela.h.machado@gmail.com

Endereço: Arthur Almeida Santos
Rua Trinta e Seis, Nº 115, Loanda, 35931-008, João Monlevade - MG, Brasil.
Endereço: Alexandre Xavier Martins
Rua Trinta e Seis, Nº 115, Loanda, 35931-008, João Monlevade - MG, Brasil.
Endereço: Marcone Jamilson Freitas Souza
Rua Trinta e Seis, Nº 115, Loanda, 35931-008, João Monlevade - MG
Endereço: Rafaela Heloisa Carvalho Machado
Km 7 - Zona Rural, MG-230, Rodoviário, 38810-000, Rio Paranaíba - MG

Editor-Chefe: Dr. Tonny Kerley de Alencar Rodrigues

Artigo recebido em 07/07/2023. Última versão recebida em 01/08/2023. Aprovado em 02/08/2023.

Avaliado pelo sistema Triple Review: a) Desk Review pelo Editor-Chefe; e b) Double Blind Review (avaliação cega por dois avaliadores da área).

Revisão: Gramatical, Normativa e de Formatação

AGENCIA DE FOMENTOS: Os autores agradecem à CAPES, ao CNPq, à FAPEMIG e à UFOP pelo apoio ao desenvolvimento deste trabalho.

RESUMO

O sucesso de uma empresa requer o bom funcionamento e a confiabilidade de seus sistemas com máquinas e equipamentos em bom estado. Para isso, é essencial um bom plano de manutenção preventiva, que tende a ficar mais complexo com o aumento do número de equipamentos e o horizonte de planejamento. O objetivo deste estudo é desenvolver algoritmos meta-heurísticos eficientes para tratar o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP). O trabalho se inicia com o desenvolvimento de uma heurística construtiva e de alocação, seguido do desenvolvimento de algoritmos de busca local e meta-heurísticos baseados em Greedy Randomized Adaptive Search Procedure (GRASP), Simulated Annealing (SA) e Iterated Local Search (ILS). O desempenho dos algoritmos desenvolvidos foi comparado entre eles e com os da literatura. Para a calibragem e validação dos algoritmos meta-heurísticos, foram resolvidas instâncias fictícias pequenas. Após a calibragem, os algoritmos meta-heurísticos foram aplicados à resolução de instâncias maiores e à real. Os experimentos mostraram que o ILS foi o algoritmo de melhor desempenho e seu resultado na instância real foi 40,5%, melhor que o apresentado na literatura.

Palavras Chave. Planejamento de Manutenção de Longo Prazo. Grasp. Simulated Annealing. Iterated Local Search. Meta-Heurísticas.

ABSTRACT

The success of a company requires the proper functioning and reliability of its systems with machines and equipment in good condition. For this, a good preventive maintenance plan is essential, which tends to become more complex as the number of equipment and the planning horizon increases. The present work aims to develop efficient meta-heuristic algorithms for the Long-Term Preventive Maintenance Scheduling Problem (PPOMPLP). The work begins with the development of a constructive and allocation heuristic, followed by the development of local search and meta-heuristic algorithms based on Greedy Randomized Adaptive Search Procedure (GRASP), Simulated Annealing (SA), and Iterated Local Search (ILS). The performance of the proposed algorithms was compared among themselves and with those of other studies in the literature. Small fictitious instances were used to calibrate and validate the meta-heuristic algorithms. After calibration, they were applied to solve larger and real instances. The experiments showed that the ILS was the best-performing algorithm, and its result for the real instance was 40.5% better than that presented in the literature.

Keywords: Long-Term Maintenance Scheduling. Grasp. Simulated Annealing. Iterated Local Search. Meta-Heuristics.

1 INTRODUÇÃO

As atividades de manutenção preventiva são essenciais para a disponibilidade e confiabilidade dos sistemas dentro da indústria. Os mais diversos equipamentos, desde esteiras transportadoras a empilhadeiras, possuem um cronograma de manutenções que devem ser realizadas com certa frequência, geralmente sugeridas pelo fabricante. As atividades de manutenção preventiva podem ser inspeção, limpeza, lubrificação, ajuste, alinhamento ou substituição de componentes desgastados (EBRAHIMIPOUR *et al*., 2015).

A manutenção preventiva é especialmente importante para evitar que ocorram falhas na operação que possam causar danos consideráveis ao sistema, como quebra de máquina, ou ao ambiente, como poluição, explosões, perda de informação (LEVITIN *et al*., 2021). Normalmente, essas atividades de manutenção preventiva necessitam de uma equipe especializada e são executadas em um equipamento específico. Isso faz com que um número limitado de atividades possa ser executado dentro de determinado período de tempo.

Para que o maior número de atividades ou para que as mais importantes sejam executadas, é necessário que se realize a programação das ordens de manutenção preventiva. Apesar da programação da manutenção preventiva ser um tema amplamente abordado, alguns estudos como Lee e Cha (2016) e Wang e Miao (2021) trabalham apenas com modelos para previsão das falhas e não com modelos focados na programação das ordens de manutenção preventiva.

Outros trabalhos como Hedjazi (2015) e Alfares (2022) tratam de modelos para a resolução de problemas de programação de ordens de manutenção. Hedjazi (2015) trabalha com um problema de programação de ordens de manutenção em máquinas em paralelo não relacionadas. Os operadores, dependendo da habilidade, realizam a manutenção em um tempo diferente. As ordens podem ser alocadas com atraso, e o objetivo é minimizar a média ponderada dos tempos de atraso. Alfares (2022) propõe um problema de sequenciamento de ordens de manutenção preventiva em máquinas em paralelo durante uma parada de máquinas para a manutenção. Nesse caso, a programação é feita de forma a encurtar o período de parada.

No entanto, esses trabalhos não tratam a programação de ordens de manutenção envolvendo o dimensionamento das equipes, as janelas de atendimento das ordens, a habilidade das equipes em executar essas ordens, a disponibilidade dos equipamentos usados para realizar essas ordens e o longo prazo de programação.

De nosso conhecimento, o único trabalho encontrado na literatura que trata dessas características é o de Aquino et al. (2019), que define o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (PPOMPLP). O PPOMPLP é considerado um problema de programação de manutenções preventivas em máquinas em paralelo, sendo discutidos pelos autores por meio de uma abordagem heurística. Os métodos heurísticos são aplicados porque esse problema é NP-difícil, uma vez que um caso particular dele, o de sequenciamento em uma máquina, é NP-difícil (QI *et al*., 1999).

Em seu estudo, Aquino et al. (2019) propuseram uma formulação de programação matemática (*Mixed Integer Linear Programming* MILP), assim como algoritmos meta-heurísticos baseados em *Simulated Annealing* (SA), *Variable Neighborhood Search* (VNS), *Multi-Start Variable Neighborhood Search* (MSVNS), *Biased Random-Key Genetic Algorithm* (BRKGA) e *Biased Random-Key Memetic Algorithm* (BRKMA) para tratar instâncias maiores do problema. No presente trabalho são apresentados, além de um algoritmo baseado em SA, algoritmos baseados em *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Iterated Local Search* (ILS), os quais têm sido usados com sucesso na resolução de vários outros problemas de otimização combinatória (ERTEM *et al*., 2022; DELORME *et al*., 2021).

Este trabalho tem por objetivo desenvolver algoritmos meta-heurísticos eficientes para tratar o PPOMPLP, aplicado a um cenário real de uma empresa mineradora, assim como em Aquino *et al*. (2019). Para tal, propõe-se inicialmente um método heurístico construtivo, composto por duas etapas: sequenciamento e alocação. No sequenciamento, define-se a sequência em que serão alocadas as ordens; na etapa de alocação, as ordens são alocadas às respectivas equipes ao longo do período de programação. A partir da solução obtida pela heurística construtiva, são desenvolvidos métodos de busca local para refinar essa solução. Esses métodos de construção e refinamento são, então, incorporados aos algoritmos meta-heurísticos GRASP, SA e ILS. Para avaliar os algoritmos desenvolvidos, são usadas instâncias reais, assim como instâncias fictícias baseadas no mesmo contexto dessas instâncias reais. Os resultados obtidos são comparados entre si e com a literatura.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta o problema de pesquisa. A Seção 3 apresenta o referencial teórico, contextualizando o problema de programação de manutenções heurísticas e meta-heurísticas. A Seção 4 apresenta a metodologia adotada no estudo. As Seções 5 e 6 apresentam as heurísticas construtivas e a heurística de alocação. Na Seção 6 apresentam-se os algoritmos meta-heurísticos utilizados,

e na Seção 8 os resultados obtidos. Por fim, na Seção 9, são apresentadas as conclusões e as sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Problema de pesquisa

O PPOMPLP considera que existem diferentes grupos de equipes de manutenção com competências diferentes. Por exemplo, um grupo de manutenção mecânica, outro de manutenção elétrica e diversos outros grupos com diferentes especialidades. Cada grupo possui um número específico de equipes com uma determinada disponibilidade de tempo, e cada equipe não pode realizar mais de uma manutenção em paralelo. As manutenções a serem realizadas possuem como parâmetros um grau de prioridade, o tempo de início e de fim em que podem ser realizadas, sendo essa uma janela de tempo em que a manutenção pode ser programada. Possui também uma duração, um tipo de manutenção, que especifica qual grupo de manutenção é capaz de realizar a manutenção e qual equipamento a equipe de manutenção deve utilizar.

A função objetivo visa à minimização do número de manutenções não realizadas. Cada ordem não alocada gera um custo de terceirização para a empresa. Esse custo está apresentado na modelagem do problema como penalidade. O valor da penalidade foi definido como sendo o tempo de duração das ordens não alocadas multiplicado pela sua prioridade.

O problema real do tipo PPOMPLP, tratado neste trabalho, vem de uma das plantas de beneficiamento de minério de ferro de uma multinacional que realiza o planejamento de todas as ordens de manutenção para o período de um ano (52 semanas). Quando a empresa realiza o plano de manutenção, cada área da manutenção (ex.: mecânica, elétrica) realiza seu próprio plano baseado apenas na disponibilidade do equipamento alvo da manutenção. Esse plano é inserido no sistema ERP (do inglês *Enterprise Resource Planning*, um sistema de gestão empresarial) utilizado pela empresa; porém, esse sistema não contém informações referentes às restrições de capacidade presentes no sistema real, como, por exemplo, a disponibilidade de mão de obra. Então, a programação das manutenções é ajustada mensalmente para cada equipe, conforme a disponibilidade de mão de obra e equipamentos necessários. Este método de planejamento de longo prazo faz com que, em geral, apenas 50% das ordens de manutenção sejam atendidas. O restante das ordens de manutenção é

terceirizado, o que resulta em mais custo para a empresa. O problema real envolve, normalmente, mais de 33000 ordens de manutenção preventiva por ano.

As técnicas de sequenciamento podem ser aplicadas para a elaboração de planos de manutenção. Problemas de sequenciamento são descritos pela alocação de *n* atividades a um número *m* de máquinas. Este tipo de problema pode ser descrito pelo trio $\alpha \mid \beta \mid \gamma$ (Pinedo, 2016). O campo α descreve o ambiente da máquina. O campo β detalha as características de processamento e as restrições. O campo γ contém os objetivos a serem otimizados. Para cada um desses campos, adota-se uma terminologia de acordo com o problema em evidência.

Essa terminologia de Pinedo (2016) foi utilizada por Aquino et al. (2019) para representar uma simplificação do PPOMPLP como um problema $P_m \mid r_j M_j \mid \gamma$. Um ponto de discordância é quanto ao campo γ, uma vez que Aquino et al. (2019) consideram que o objetivo de otimização seria minimizar a mão de obra necessária para executar o maior número de tarefas, não havendo notação correspondente, o campo permanece γ. Porém, o problema em questão possui dois objetivos de otimização: minimizar o custo relacionado às ordens de manutenção não atendidas e minimizar o custo de mão de obra. O custo de mão de obra tem ordem de grandeza muito menor comparado ao custo de cada manutenção não realizada; portanto, o segundo pode ser considerado o principal objetivo de otimização.

Seguindo essa mesma terminologia de Pinedo (2016), o presente trabalho aborda um problema $P_m \mid r_j M_j \mid \sum_j w_j T_j$. O campo P_m refere-se a problemas com *m* máquinas paralelas, onde r_j significa que a atividade *j* não pode começar seu processamento antes da data de lançamento. M_j é utilizado para problemas de máquinas paralelas, nos quais nem todas *m* máquinas são capazes de processar todas as atividades. O último campo, $\sum_j w_j T_j$ se refere à média ponderada dos tempos de atraso. Apesar de as manutenções não poderem ser alocadas com atraso, as ordens não alocadas entram como custo para a função objetivo, de forma ponderada, de acordo com sua prioridade.

Problemas de sequenciamento de máquinas em paralelo têm sido estudados há décadas. Alidaee e Rosa (1997) estão entre os primeiros a publicarem uma abordagem heurística para resolução desse tipo de problema. Porém, os autores tratam apenas do sequenciamento da produção. Problemas de sequenciamento de produção e de manutenção têm sido tratados de forma diferente, tanto na literatura quanto na indústria. Apesar de serem problemas semelhantes, as programações podem ser conflitantes porque a programação de manutenções preventivas interfere na programação de produção e para a realização da manutenção preventiva, o equipamento deve cessar a produção (Avalos-Rosales et al., 2018).

A Tabela 1 apresenta os principais estudos que tratam de programação de manutenções preventivas em máquinas em paralelo por meio de uma abordagem heurística. Os estudos foram classificados em relação ao tipo de máquina e de programação. Entre os estudos citados, a maioria foca na programação de produção em conjunto com as manutenções. Além disso, em relação aos problemas de máquinas em paralelo, parte dos trabalhos trata de um subtipo, que são as máquinas em paralelo não relacionadas. Em relação ao "Objetivo a minimizar", são apresentados quais os indicadores de desempenho foram utilizados na função objetivo. A maioria tem como objetivo minimizar o *makespan*, que é o tempo total de processamento.

Tabela 1 – Revisão De Literatura.

Autores	**Máquinas**	**Tipos de programação**	**Objetivo a minimizar**	**Abordagem**
HEDJAZI (2015)	paralelas não relacionadas	manutenção	média ponderada dos tempos de atraso	heurística construtiva
FAKHER; NOURELFATH; GENDREAU (2016)	paralelas	produção e manutenção	custos	algoritmo genético e *tabu search*
RUIZ-TORRES; PALETTA; M'HALLAH (2016)	idênticas em paralelo	produção e manutenção	*makespan*	heurísticas construtivas
UPASANI et al. (2017)	idênticas em paralelo	manutenção	custos	algoritmo memético e *particle swarm optimization*
AVALOS-ROSALES et al., (2018)	paralelas não relacionadas	produção e manutenção	*makespan*	multi-start
FU et al. (2019)	idênticas em paralelo	produção e manutenção	*makespan*	*three-level particle swarm optimization e VNS*
LIU et al. (2021)	paralelas sincronizadas	produção e manutenção	custo	*cooperative co-evolutionary genetic algorithm*
ALFARES; MOHAMMED; GHALEB (2021)	idênticas em paralelo	produção e manutenção	*makespan*	VNS
DELORME; IORI; MENDES (2021)	paralelas não relacionadas	produção e manutenção	*makespan*	ILS
LU et al. (2021)	paralelas	produção e manutenção	*makespan*	VNS e *discrete black hole*
ALFARES (2022)	paralelas	manutenção	*makespan*	heurística construtiva
ERTEM et al. (2022)	paralelas	manutenção	*makespan*	GRASP

Na Tabela 1, a coluna “Abordagem” indica quais abordagens heurísticas foram aplicadas para resolução do problema de programação de manutenções preventivas em máquinas em paralelo. Foram aplicadas abordagens variadas para esse tipo de problema. As heurísticas mais aplicadas foram as heurísticas construtivas e a VNS. Em relação às meta-heurísticas, GRASP, SA e ILS, focos deste estudo, destaca-se o estudo de Delorme *et al.* (2021), com a aplicação do ILS, e Ertem *et al*. (2022), com a aplicação do GRASP. Não foram destacados estudos para o problema de programação de manutenções preventivas em máquinas em paralelo aplicando a meta-heurística SA.

O ILS é amplamente aplicado à resolução de problemas de programação de produção, que é um problema análogo ao problema de programação de ordens de manutenção preventiva (XU *et al.*, 2019). Delorme *et al*. (2021) abordam um problema de sequenciamento de ordens de produção e manutenções preventivas em máquinas em paralelo não relacionadas. O objetivo é minimizar o *makespan*. Os autores utilizam o MILP proposto por Ruiz-Torres et al. (2016) e propõem quatro novos modelos matemáticos a partir de diferentes pontos de vista do problema. Os autores propõem também um algoritmo meta-heurístico ILS para resolução do problema. Dentre os modelos exatos, uma formulação baseada em fluxo de arco obteve melhor desempenho. O método ILS teve resultados competitivos para as instâncias grandes do problema.

Outra técnica de rápida implementação e bons resultados é o GRASP. O GRASP é uma meta-heurística *multi-start* que consiste de duas fases: a fase construtiva, na qual uma solução viável é produzida, e a segunda fase, que é um método de busca local, que busca uma solução ótima local (FEO; RESENDE, 1995). Ertem *et al*. (2022) abordam um problema de sequenciamento de ordens de manutenção preventiva com máquinas em paralelo. O objetivo é minimizar o *makespan* que, nesse caso, é equivalente a minimizar o número de tarefas em atraso. Como método de resolução, são propostos dois modelos MILP para resolução das instâncias pequenas e calibragem das heurísticas. Dois métodos heurísticos construtivos foram propostos, sendo o primeiro dependente de uma solução inicial gerado por um método exato e o segundo que utiliza um método de exploração das vizinhanças semelhante ao GRASP. Apesar de as heurísticas não atingirem valores ótimos em todas as instâncias pequenas, como o MILP, elas apresentam melhor desempenho nas instâncias grandes. A segunda heurística apresenta resultados melhores que a primeira.

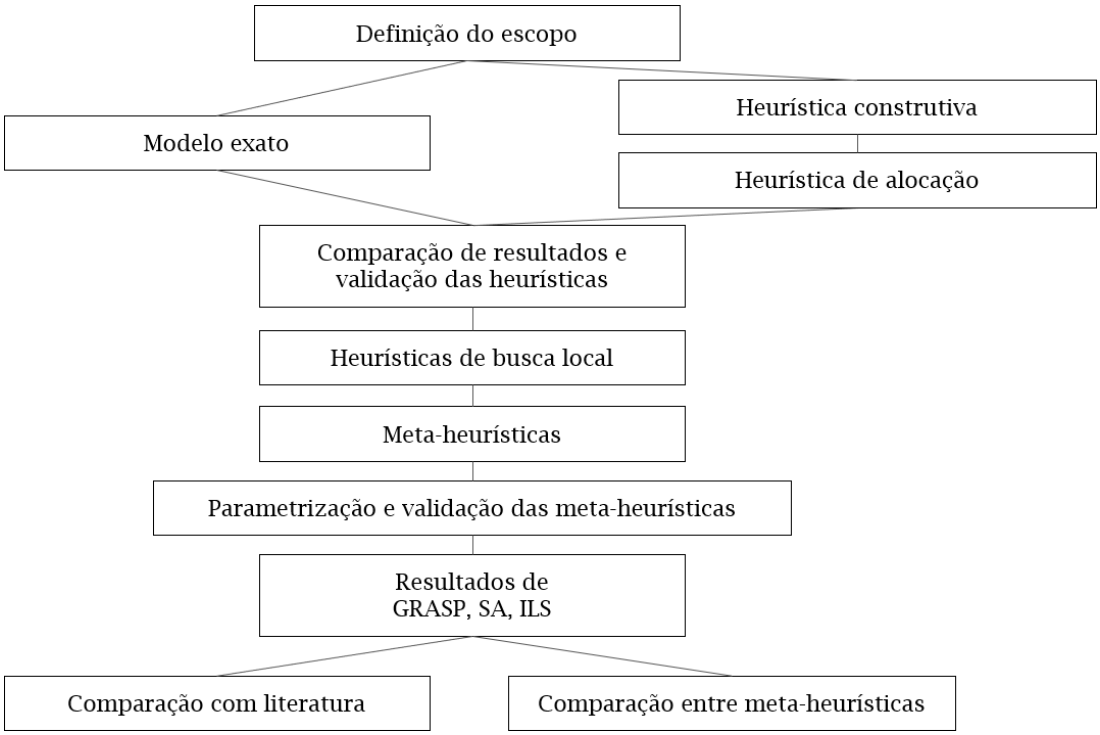
O conceito de SA foi aplicado pela primeira vez para resolução de problemas de otimização por Kirkpatrick *et al*. (1983). O método é uma meta-heurística inspirada no processo físico de resfriamento lento de um material, no qual a solução atual é modificada

iterativamente com base em movimentos aleatórios e aceitação de soluções piores, de acordo com uma probabilidade determinada pela temperatura. Ao longo do tempo, a temperatura diminui, levando o algoritmo a convergir para uma solução ótima ou aproximadamente ótima.

3 METODOLOGIA

Esta pesquisa segue as etapas conforme esquema da Figura 1, que se inicia pela definição do escopo da pesquisa. Em sequência, é realizado o desenvolvimento da heurística construtiva, que produz uma programação inicial. Com a sequência definida pela heurística construtiva, a heurística de alocação é utilizada para estruturar as atividades de cada equipe ao longo do período de programação. Os resultados dessa heurística construtiva foram comparados aos resultados do modelo exato, para sua validação.

Figura 1 – Esquema da metodologia



Posteriormente, foram desenvolvidos os métodos de busca local *first improvement* e *random descent*. Esses métodos heurísticos são subordinados aos algoritmos meta-heurísticos. Portanto, a etapa seguinte é a implementação dos algoritmos meta-heurísticos GRASP, SA e ILS para resolução do PPOMPLP. A parametrização desses algoritmos utiliza

o pacote IRACE (López-Ibáñez *et al.*, 2016), realizando rodadas teste com uma amostra das instâncias fictícias. Os valores dos parâmetros testados para cada algoritmo meta-heurístico estão na Tabela 2. A presença do valor "NA" no campo indica que o parâmetro não se aplica ao algoritmo meta-heurístico.

Tabela 2 – Valores dos parâmetros utilizados no IRACE

Algoritmo	α	Número máximo de iterações	T_0	Níveis de perturbação
GRASP	{0,05; 0,1; 0,2; 0.9}	{500, 2000, 10000}	NA	NA
SA	{0,5; 0.85; 0,9; 0,95}	{500, 2000, 10000, 20000}	{10, 25, 100, 1000}	NA
ILS	NA	{500, 2000, 10000, 20000}	NA	{2, 3, 4}

Para a validação dos algoritmos meta-heurísticos, são resolvidas instâncias fictícias pequenas. Após a calibragem e validação, os algoritmos foram aplicados à resolução de instâncias maiores e a real. Os resultados obtidos foram comparados entre os algoritmos desenvolvidos e com os resultados de outros algoritmos apresentados na literatura.

3.1 Formulação matemática

O modelo matemático utilizado foi o desenvolvido por Aquino et al. (2019). Para a descrição matemática do modelo foram utilizadas as seguintes notações. $T = \{1, 2, ..., n\}$ é o conjunto de n atividades de manutenção que devem ser realizadas pelo conjunto $W = \{1, 2, ..., m\}$ de m equipes de trabalho. Cada manutenção $i \in T$ é associada ao conjunto $W_i \subseteq W$ de equipes de trabalho capazes de realizá-la. Também associados a cada manutenção $i \in T$ estão o tempo p_i necessário para executá-la, A_i o equipamento em que a manutenção será realizada, e a janela de tempo (e_i, l_i) que a manutenção pode ser realizada. A penalidade por não realizar a manutenção é w_i. O valor da penalidade foi definido como o tempo p_i, multiplicado por um valor de prioridade da ordem de manutenção definido pela empresa.

Cada equipe de trabalho $k \in W$ está disponível no período $(0, h_k)$. O conjunto $T_k \subseteq T$ indica as manutenções que podem ser realizadas pela equipe de trabalho $k \in W$. Para o modelo de programação linear inteira mista (MILP), foram definidas as variáveis de decisão do modelo:

- $x_{ij}^{k} = 1$, se a manutenção i precede imediatamente a manutenção j executada pela equipe de trabalho k; 0, caso contrário;
- $y_{ik} = 1$, se a manutenção i será executada pela equipe de trabalho k; 0, caso contrário;
- z_k = 1, se a equipe de trabalho k é utilizada; 0, caso contrário;
- $c_{ik} \geq 0$, tempo de conclusão da manutenção i pela equipe de trabalho k;
- r_{ij} = 1, se a manutenção i precede imediatamente a manutenção j para o caso em que ambas as manutenções se referem ao mesmo equipamento ($A_i = A_j$) e que ambas são executadas por equipes diferentes; 0, caso contrário.

O modelo de programação matemática que define o PPOMPLP é, então, apresentado pelas Equações (1) a (17):

$$min \sum_{k \in W} z_k + \sum_{k \in \mathcal{T}_k} w_i \left(1 - \sum_{k \in W_i} y_{ik}\right) \quad (1)$$

Sujeito às restrições:

$$\sum_{k \in W_i} y_{ik} \leq 1 \quad \forall i \in \mathcal{T} \quad (2)$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\}\{j\}} x_{ij}^{k} = y_{ik} \quad \forall j \in \mathcal{T}, k \in \mathcal{W}_j \quad (3)$$

$$\sum_{j \in \mathcal{T}_k} x_{0j}^{k} = z_k \quad \forall k \in \mathcal{W} \quad (4)$$

$$\sum_{i \in \mathcal{T}_k \cup \{0\}\{l\}} x_{il}^{k} = \sum_{j \in \mathcal{T}_k \cup \{0\}\{l\}} x_{lj}^{k} \quad \forall k \in \mathcal{W}, l \in \mathcal{T}_k \quad (5)$$

$$c_{0k} = 0 \quad \forall k \in \mathcal{W} \quad (6)$$

$$c_{jk} \geq c_{ik} + p_j - M'_{ij} * \left(1 - x_{ij}^{k}\right) \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \quad (7)$$

$$c_{ik} \geq (e_i + p_i) * y_{ik} \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \quad (8)$$

$$c_{ik} \leq l_i \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \quad (9)$$

$$c_{jk'} \leq c_{ik} - p_i + M''_{ij} * r_{ij} \quad \forall k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, | \, k \neq k', i < j, A_i = A_j \quad (10)$$

$$c_{jk'} \leq c_{ik} + p_i - M'_{ij} * (1 - r_{ij}) \quad \forall k \in \mathcal{W}, k' \in \mathcal{W}, i \in \mathcal{T}_k, j \in \mathcal{T}_{k'}, | \, k \neq k', i \prec j, A_i = A_j \quad (11)$$

$$c_{ik} - M * (1 - y_{ik}) \leq h_k \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \quad (12)$$

$$x_{ij}^k \in \{0,1\} \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\}, j \in \mathcal{T}_k \cup \{0\} \quad (13)$$

$$y_{ik} \in \{0,1\} \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \cup \{0\} \quad (14)$$

$$z_k \in \{0,1\} \quad \forall k \in \mathcal{W} \quad (15)$$

$$c_{ik} \geq 0 \quad \forall k \in \mathcal{W}, i \in \mathcal{T}_k \quad (16)$$

$$r_{ij} \in \{0,1\} \quad \forall i \in \mathcal{T}, j \in \mathcal{T} \mid i \prec j, A_i = A_j \quad (17)$$

A Equação (1) define a função objetivo, que busca minimizar a soma do custo de mão de obra com as manutenções não atendidas. O conjunto de restrições (2) garante que cada manutenção será executada por, no máximo, uma equipe de trabalho. O conjunto de restrições (3) garante que toda manutenção executada terá uma mão de obra alocada. As restrições (4) garantem que uma determinada equipe de trabalho só será alocada se ela for utilizada para execução de alguma manutenção. As restrições (5) garantem o sequenciamento das ordens de manutenção executadas por cada equipe de trabalho. O conjunto de restrições (6) assegura que o tempo de conclusão de uma manutenção fictícia, criada para facilitar a modelagem, é nulo para todas as equipes de trabalho. As restrições (7), (10) e (11) garantem que a equipe de trabalho não pode executar mais de uma ordem de manutenção simultaneamente, e cada equipamento não pode ter mais de uma ordem de manutenção sendo executada simultaneamente. Mais especificamente, as restrições (7) são ativadas apenas quando a manutenção *j* for realizada imediatamente após a manutenção *i* pela equipe *k*. Por sua vez, as restrições (10) são ativadas quando a manutenção *j* precede imediatamente a manutenção *i* para o caso em que ambas as manutenções se referem ao mesmo equipamento ($A_i = A_j$) e executadas por equipes diferentes. As restrições (11) são similares às (10) quando a manutenção *i* precede imediatamente à manutenção *j*. Para ativar ou desativar essas restrições, as constantes M' e M'' devem assumir valores maiores ou iguais a $l_i + p_j$ e $l_j + p_i$, respectivamente. As restrições (8) e (9) garantem que cada manutenção seja executada dentro de sua janela de tempo, enquanto as restrições (12)

garantem que o número de horas alocadas para uma equipe de trabalho seja menor ou igual ao número de horas disponíveis. Por fim, as restrições (13)-(17) indicam o domínio e escopo das variáveis de decisão.

Importante destacar que esse modelo matemático necessita da criação de uma manutenção fictícia para seu funcionamento. Os valores que são inseridos nos parâmetros dessa manutenção fictícia não importam, mas, como boa prática, são utilizados valores incomuns às demais manutenções. Ao fazer a leitura dos dados, essa manutenção será o primeiro elemento ($i = 0$) e as demais manutenções são enumeradas na sequência a partir dos índices $i = \{1, 2, ..., n\}$.

a Testes com o modelo e relaxações

Ao longo da testagem do modelo exato, foram realizadas rodadas teste considerando a relaxação de cada uma das variáveis de decisão do modelo. Para a comparação do processamento relaxado com os resultados do processamento não relaxado, foram utilizadas apenas as instâncias que tiveram o processamento concluído com menos de uma hora. Isso garante que a comparação seja feita apenas entre resultados ótimos. Os resultados obtidos estão apresentados na Tabela 3.

Tabela 3 – Resultados dos testes do modelo matemático com variáveis relaxadas

Instância				Solução não	Solução com variável relaxada							
ID	*n*	*m*	A_i	relaxada	*x*	*gap*	*r*	*gap*	*y*	*gap*	*z*	*gap*
1	20	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
2	20	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
3	20	4	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
4	20	5	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
5	20	10	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
6	30	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
7	30	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
8	30	4	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
11	40	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
12	40	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
16	60	2	2	434	0	100,00%	434	0,00%	434	0,00%	434	0,00%
17	60	3	2	219	219	0,00%	3	98,63%	219	0,00%	219	0,00%
26	20	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%
27	20	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
28	20	5	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
29	20	6	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
31	30	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%
32	30	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
33	30	5	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
36	40	3	3	579	0	100,00%	579	0,00%	579	0,00%	579	0,00%

37	40	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
42	60	4	3	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
51	20	3	4	723	0	100,00%	723	0,00%	723	0,00%	723	0,00%
52	20	4	4	220	219	0,45%	4	98,18%	220	0,00%	220	0,00%
76	20	4	5	724	0	100,00%	724	0,00%	724	0,00%	724	0,00%
77	20	5	5	221	219	0,90%	5	97,74%	221	0,00%	221	0,00%

Na Tabela 3, o *gap* representa a diferença entre o resultado obtido por meio da relaxação e o resultado não relaxado. Quanto maior o *gap*, mais distantes são as soluções obtidas por meio da relaxação.

A relaxação da variável *x* resulta em soluções inviáveis, sem sequer um sequenciamento lógico. Relaxar a variável *r* produz soluções que não respeitam a restrição de que um equipamento não possa ser utilizado por mais de uma equipe ao mesmo tempo. Relaxar as variáveis *z* e *y* resulta em soluções viáveis, porém, na média não há ganho no tempo de processamento, nem nos resultados obtidos.

3.2 Heurísticas de construção e alocação

Esta seção está organizada como segue: na subseção 6.1 são apresentadas duas heurísticas construtivas para gerar uma solução inicial para o problema. Por sua vez, a subseção 6.2 apresenta um algoritmo para alocar as ordens de manutenção geradas pelas heurísticas de construção e, assim, avaliar a solução gerada.

a. Heurísticas de construção

Foram propostas duas heurísticas construtivas diferentes: a primeira chamada de “construção do sequenciamento simples das ordens” e a segunda de “construção do sequenciamento das ordens por equipamento”. A primeira, apresentada pelo Algoritmo 1, realiza o sequenciamento das ordens, tendo o tempo limite da ordem (l_c) como o principal parâmetro e, em caso de empate entre duas ordens, a penalidade (w_c) é o critério de desempate. Ordens com menor l_c e maior w_c têm prioridade de alocação. O algoritmo retorna a sequência de ordens *N*.

Algoritmo 1: Construção do sequenciamento simples das ordens

```
Entrada: Lista de ordens de manutenção
Saída: N
1  início
2      N ← ∅;
3      C ← Inicializa conjunto de ordens candidatas;
4      enquanto C ≠ ∅ faça
5          melhor_tempo ← ∞;
6          melhor_penalidade ← −1;
7          para c ∈ C faça
8              se (l_c < melhor_tempo) ou (l_c = melhor_tempo e
                w_c > melhor_penalidade) então
9                  c* ← c;
10                 melhor_tempo ← l_c;
11                 melhor_penalidade ← w_c;
12             fim
13         fim
14         N ← N ∪ {c*};
15         C ← C \ {c*};
16     fim
17 fim
```

O segundo algoritmo construtivo realiza primeiro uma avaliação de qual é o equipamento alvo com maior número de manutenções. O sequenciamento das ordens realizadas no equipamento mais utilizado é definido primeiro, seguindo os mesmos critérios do Algoritmo 1. Ordens com menor l_c e maior w_c têm prioridade de alocação. Após sequenciar todas as ordens do equipamento mais utilizado, é realizado o sequenciamento do segundo mais utilizado, até que o sequenciamento de todos seja concluído.

Para exemplificar, uma instância fictícia foi apresentada no Quadro 1. Para a instância em questão, a sequência resultante do Algoritmo 1 é $N = \{1,4,6,2,5,3\}$. O primeiro elemento dessa sequência é a ordem 1, porque tem-se o tempo limite da ordem (l_c) menor, com o valor 4. O segundo elemento (ordem 4) é o segundo da sequência por ter o segundo menor valor l_c. Para o terceiro elemento há empate do valor de l_c entre as ordens 2 e 6. Como a ordem 6 tem maior penalidade, ela ocupa a posição, seguido da ordem 2 que será o quarto elemento. Os dois últimos elementos da sequência são posicionados de acordo com os respectivos valores de l_c, já que não há empate.

Quadro 1 – Características das ordens de manutenção.

Tarefa de manutenção		Equipamento	Especialidade	Início	Fim	Duração	Penalidade
1.	Alinhamento	Carro	Mecânica	0	4	1	20
2.	Alinhamento	Caminhão	Mecânica	2	7	2	30
3.	Revisão de motor	Carro	Mecânica	3	9	3	40
4.	Revisão elétrica	Moto	Elétrica	2	6	1	20
5.	Revisão elétrica	Carro	Elétrica	3	8	2	30
6.	Revisão de motor	Caminhão	Mecânica	4	7	1	40

Fonte: adaptado de Aquino *et al*. (2019)

Para o segundo algoritmo construtivo, como o equipamento “carro” é o mais utilizado, a sequência se inicia por ele. Isso resulta na seguinte sequência: $N = \{1,5,3,6,2,4\}$. Para a construção desta sequência, avalia-se para as primeiras posições apenas ordens executadas no equipamento carro (ordens 1, 3 e 5). Portanto, o primeiro elemento da sequência é a ordem 1, com menor l_c, seguido da ordem 5, com segundo menor l_c e, por último, a ordem 3. O segundo equipamento mais utilizado é o caminhão (ordens 2 e 6). As ordens 2 e 6 têm valores de l_c iguais, porém a penalidade da ordem 6 é maior, fazendo com que ela seja o quarto elemento da sequência, seguida pela ordem 2. O último equipamento é a moto, com apenas a ordem 4, que é então o último elemento da sequência. Os algoritmos meta-heurísticos desenvolvidos utilizam o algoritmo construtivo com melhor resultado como solução inicial do problema, porque há diferença de acordo com a instância.

b. Heurística de alocação

Com a sequência das ordens definida por um dos dois algoritmos de construção apresentados na seção anterior, é necessário realizar a sua alocação. A alocação é realizada por equipes, iniciando pela equipe ou grupo de equipes mais ocupado. Para verificar qual é a equipe mais ocupada, primeiro soma-se a duração de todas as atividades que podem ser realizadas por um grupo de equipes (no exemplo do Quadro 1, o grupo de equipes contém apenas as especialidades mecânica ou elétrica). A duração total é, então, dividida pelo número de equipes em um grupo. O grupo com maior ocupação tem prioridade na alocação das ordens.

Para a alocação das ordens, foi implementado o Algoritmo 2. A alocação das ordens é realizada do começo da janela de tempo da ordem c (e_c). Caso não seja possível

alocar a ordem na primeira posição, o horário de início da ordem é postergado até ser possível. Caso ultrapasse a janela de tempo da ordem, a ordem será alocada a outra equipe. Caso não seja possível alocá-la em nenhuma das outras equipes, a ordem não é alocada e entra como penalidade no resultado da função objetivo. Há dois motivos para que a ordem não possa ser alocada em sua primeira posição: caso já exista uma ordem alocada na mesma equipe no mesmo horário ou caso tenha uma ordem alocada no mesmo equipamento em outra equipe no mesmo horário. As variáveis *início* e *fim* utilizadas no Algoritmo 2 são os horários de início e fim de execução de uma ordem na programação. O algoritmo retorna à solução *s*, à penalidade total e à lista de ordens não utilizadas.

A Figura 2 exemplifica o funcionamento da alocação de ordens. O primeiro gráfico de Gantt representa a alocação das ordens do Quadro 1, seguindo o método "construção do sequenciamento simples das ordens" (Algoritmo 1). A sequência N = {1,4,6,2,5,3} deve começar pelas equipes de mecânica. Isso pode ser traduzido em uma sequência menor: N' = {1,6,2,3}. A janela para alocação da ordem 1 tem os valores (0, 4). Como não há nenhuma ordem alocada no instante 0, a ordem 1 é alocada nesse instante. A ordem 6 tem janela (4, 7) e pode ser alocada no instante 4. A ordem 2 tem janela (2, 7) e pode ser alocada no instante 2. A ordem 3 tem janela (3, 9), porém o instante 3 está ocupado pela ordem 3. O instante 4 está ocupado pela ordem 6. Portanto, a melhor alocação para a ordem 3 é no instante 5.

Figura 2 – Exemplo de alocação de ordens

Sequência Simples: N = {1,4,6,2,5,3} Sequência Mecânica: N' = {1,6,2,3} Sequência Elétrica: N'' = {4,5}

Equipes\período	0	1	2	3	4	5	6	7	8	9
Mecânica A	1		2	2	6	3	3	3		
Mecânica B										
Elétrica			4	5	5					

Ferramentas mais usadas: Carro > Caminhão > Moto
Sequência Equipamentos: N = {1,5,3,6,2,4} Sequência Mecânica: N' = {1,3,6,2} Sequência Elétrica: N'' = {5,4}

Equipes\período	0	1	2	3	4	5	6	7	8	9
Mecânica A	1			3	3	3	6	2	2	
Mecânica B										
Elétrica			4				5	5		

Legenda dos Equipamentos: Carro, Caminhão, Moto

Após a alocação de todas as ordens das equipes de mecânica, a alocação é realizada para as equipes de elétrica, seguindo o restante da sequência $N'' = \{4,5\}$. A ordem 4 tem janela (2, 6) e pode ser alocada no instante 2. A ordem 5 tem janela (3, 8) e pode ser alocada no instante 3. O mesmo procedimento de alocação é realizado para a "construção do sequenciamento das ordens por equipamento" (segundo Gantt), sendo apenas a sequência diferente.

Algoritmo 2: Aloca ordens

Entrada: Lista de ordens de manutenção, lista de equipes, sequência(N);
Saída: s, penalidade, ordens não utilizadas;

```
1  início
2      s ← ∅;
3      Organiza equipes, começando pelas equipes mais ocupadas;
4      para k ∈ W faça
5          C ← Inicializa conjunto de ordens candidatas, seguindo sequência N, apenas
             com ordens que a equipe é capaz de realizar e com ordens ∉ s;
6          enquanto C ≠ ∅ e fim_c ≤ h_k faça
7              c ← C[0];
8              inicio_c ← e_c;
9              fim_c ← inicio_c + p_c;
10             enquanto Houver conflito de horários entre c e s[i] e fim_c ≤ l_c e
                 fim_c ≤ h_k faça
11                 para i ∈ s faça
12                     se Ordem s[i] alocada na mesma equipe e mesmo horário então
13                         inicio_c ← s[i][fim_c];
14                         fim_c ← inicio_c + p_c;
15                     fim
16                     se Ordem s[i] no mesmo equipamento, alocada em outra equipe e
                         mesmo horário então
17                         inicio_c ← s[i][fim_c];
18                         fim_c ← inicio_c + p_c;
19                     fim
20                 fim
21             fim
22             se fim_c > l_c ou fim_c > h_k então
23                 C ← C \ {c};
24             senão
25                 Insere c na solução s, com horário de início e término definidos por
                     inicio_c e fim_c respectivamente;
26                 C ← C \ {c};
27             fim
28         fim
29     fim
30     Calcula penalidade e ordens não utilizadas;
31 fim
```

3.3 Algoritmos meta-heurísticos

Os valores dos parâmetros utilizados nos algoritmos meta-heurísticos foram obtidos por meio de calibração utilizando o pacote IRACE. O único parâmetro fixo e comum a todos os algoritmos meta-heurísticos é o tempo limite de execução, fixado em *n* segundos tal como em Aquino *et al*. (2019), sendo *n* o número de ordens de manutenção da instância. Cada algoritmo foi executado com cinco repetições para cada instância. O algoritmo GRASP foi desenvolvido com base no de Feo e Resende (1995), utilizando como método de busca local o *random descent*. O *random descent* desenvolvido realiza trocas aleatórias entre duas ordens (*i, j*) da sequência dada *N*. Só é avaliada a troca das ordens *i* e *j* se elas estiverem dentro de um mesmo período de programação ($e_j > l_i > l_j$ ou $e_i > l_j > l_i$). Caso a troca produza uma solução melhor, ela é aceita, modificando-se *N*.

Para cada nova sequência gerada dentro do algoritmo meta-heurístico é necessário refazer a alocação das ordens para que seja calculado o novo custo da função objetivo. A heurística de alocação, por ser a mais custosa em termos de processamento, tornaria inviável que todo o período de programação fosse realocado a cada troca. Por isso, a cada nova sequência gerada dentro dos métodos meta-heurísticos, é refeita apenas a alocação dentro de períodos em que haja ordens não alocadas, mantendo a alocação anterior para os demais períodos.

O algoritmo SA foi desenvolvido com base no trabalho de Kirkpatrick et al. (1983). O vizinho aleatório é gerado trocando-se duas ordens aleatórias (*i, j*) da sequência de programação. Para limitar o número de trocas, as ordens *i* e *j* só são trocadas se estiverem dentro do mesmo período de programação, semelhante à lógica utilizada no *random descent*. Ordens em períodos de tempos diferentes têm pouca influência na alocação da outra.

O algoritmo ILS foi adaptado daquele proposto por Lourenço et al. (2019). Como métodos de busca local foram utilizados o *random descent*, já explicado anteriormente, e o *first improvement*, sendo sorteados, com igual probabilidade, qual será utilizado em cada iteração. O método *first improvement* realiza a troca das ordens *i* e *j*, começando por *i*, como primeiro elemento de *N*, e *j*, como segundo elemento. A cada iteração é acrescida uma unidade à *i* ou *j*, até que todas as trocas possíveis sejam avaliadas. Ao longo das iterações, se uma troca representa melhora no resultado da função objetivo, a troca é realizada, atualizando-se *N*, e reiniciando-se o método *first improvement*. As trocas do ILS também só são avaliadas se estiverem dentro de um mesmo período de programação.

4 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados da comparação entre os três algoritmos desenvolvidos neste trabalho. Primeiramente em relação a todas as instâncias e, posteriormente, em relação a um agrupamento por tamanho. Em seguida é verificado o desvio-padrão médio dos algoritmos e se há diferença estatística entre eles. Por fim, os algoritmos desenvolvidos são comparados com os da literatura.

Os algoritmos heurísticos e meta-heurísticos foram todos desenvolvidas em Python, versão 3.9. Cada algoritmo meta-heurístico foi parametrizado com o tempo de processamento igual a *n* segundos, sendo *n* o número de ordens de manutenção da instância e com cinco repetições para cada instância.

Todos os algoritmos foram executados em um computador Intel(R) Core (TM) i7-4790 CPU @ 3.60GHz, 15GB de memória RAM sob o sistema operacional Ubuntu 16.04.6 LTS (Xenial Xerus). O computador pertence ao Laboratório de Simulação e Otimização de Sistemas (LASOS) da Universidade Federal de Ouro Preto (UFOP). Vale destacar que o presente trabalho não utiliza o processamento em *threads* paralelas.

A Tabela 4 apresenta o desvio-padrão médio do *gap* das instâncias. O *gap* foi calculado pela diferença percentual entre o melhor valor obtido pelo algoritmo e o melhor valor disponível na literatura. O ILS apresenta um desvio-padrão médio menor dentre os algoritmos meta-heurísticos aplicados, o que indica que essa é a implementação com a melhor repetibilidade dentre as três.

Tabela 4 – Desvio-padrão médio dos algoritmos GRASP, SA e ILS

	GRASP	SA	ILS
Todas as instâncias	10,6%	7,6%	7,1%

Para análise dos resultados, as instâncias foram divididas em grupos, tal como em Aquino et al. (2019):

- P: instâncias de 20 a 80 ordens de manutenção;
- M: instâncias de 150 a 600 ordens de manutenção;
- G: instâncias de 1200 a 4800 ordens de manutenção;
- GG: instâncias de 9600 a 33484 ordens de manutenção;

A instância real foi agrupada junto às instâncias do grupo GG. A Tabela 5 apresenta as instâncias agrupadas por tamanho para análise. A coluna "Médias" mostra a média dos valores médios das repetições obtidos para cada tamanho de instância. A coluna "Melhores" apresenta a média dos menores valores obtidos dentre todas repetições para cada tamanho de instância. A coluna "Contagem melhores" indica em quantas instâncias de um mesmo tamanho cada algoritmo meta-heurístico obteve o melhor resultado.

Tabela 5 – Comparativo de resultados entre os algoritmos meta-heurísticos GRASP, SA e ILSZ

	Número de ordens	**Número de instâncias**	**GRASP**			**SA**			**ILS**		
			Médias	Melhores	Contagem melhores	Médias	Melhores	Contagem melhores	Médias	Melhores	Contagem melhores
P	20	20	**299**	**299**	**20**	**299**	**299**	**20**	**299**	**299**	**20**
	30	20	**299**	**299**	**20**	**299**	**299**	18	**299**	**299**	**20**
	40	20	332	331	19	339	332	14	**330**	**324**	**20**
	60	20	504	493	18	502	486	12	**482**	**475**	**19**
	80	20	1.345	1.315	9	1.345	1.304	7	**1.290**	**1.282**	**20**
M	150	6	891	**889**	**6**	**889**	**889**	5	**889**	**889**	**6**
	300	6	1.387	1.371	2	**1.371**	**1.370**	4	1.377	**1.370**	**5**
	600	6	3.903	3.815	1	3.867	3.803	2	**3.826**	**3.798**	**5**
G	1.200	6	9.311	9.002	1	9.047	8.916	3	**8.801**	**8.669**	**4**
	2.400	6	18.148	17.770	1	17.543	**17.025**	3	**17.531**	17.107	**4**
	4.800	6	38.389	**36.166**	**3**	38.420	37.810	1	**38.305**	37.692	2
GG	9.600	1	31.664	30.451	0	30.199	30.009	0	**30.011**	**30.002**	**1**
	19.200	1	74.257	70.963	0	**69.200**	**68.711**	**1**	69.672	68.720	0
	33.484	1	134.610	**128.856**	**1**	**133.291**	129.082	0	134.507	130.906	0

De acordo com os dados da Tabela 5, o ILS obteve resultados melhores em 99% das instâncias P, sendo o GRASP melhor em apenas uma instância e SA atingindo resultados no máximo iguais aos demais. Entre SA e GRASP, os resultados foram muito próximos na média, com SA ligeiramente melhor entre as instâncias M e G. Para as instâncias M, G e GG o ILS também foi a meta-heurística que obteve melhores resultados dentre as três, sendo melhor em 71% das instâncias M, G e GG. SA obteve o melhor resultado em 50% e o GRASP em 36,8%. Estes números somados são maiores que 100%, porque pode haver empate entre as meta-heurísticas em uma ou mais instâncias. O melhor resultado para a instância real foi obtido pelo GRASP.

Para verificar se os algoritmos meta-heurísticos possuem uma diferença estatisticamente significativa entre si, foi aplicado o teste de Friedman. O teste Friedman

(Friedman, 1940) pode ser utilizado para analisar a presença de diferenças estatisticamente relevantes entre três ou mais modelos de dados ordinais. Caso o *p-valor* resultante do teste seja maior que o nível de significância, usualmente estabelecido em 0,05, a hipótese nula de que as amostras são estatisticamente diferentes é descartada. Nesse caso, pode-se aplicar um teste *post-hoc* como o teste Nemenyi para verificar entre quais pares de medidas há diferença significativa (SHELDON *et al.*, 1996).

Ambos os testes utilizam como parâmetro de desempenho a classificação do algoritmo meta-heurístico em cada aplicação. Dessa forma, são definidas classificações (*ranks*) para cada heurística em cada instância. O teste de Friedman considera que se o *rank* médio de um modelo é aproximadamente igual a outro, então os modelos seriam estatisticamente iguais. Já o teste de Nemenyi considera que o *rank* médio de um algoritmo deve apresentar no mínimo uma certa distância em valor, definida como diferença crítica, do *rank* médio de outro algoritmo para serem considerados diferentes (WANG *et al*., 2018).

Os testes de Friedman junto ao teste *post-hoc* Nemenyi têm sido utilizados para a comparação de resultados de meta-heurísticas (MA *et al*., 2022). Como vantagem, ao considerar apenas os *ranks*, os testes se tornam menos suscetíveis a *outliers* (BROWN E MUES, 2012). Porém, ao utilizarem medidas diferentes, os testes podem apresentar resultados diferentes. Além disso, ao utilizarem apenas o *rank*, a dimensão de diferença entre os algoritmos meta-heurísticos não é considerada na análise. Dessa forma, a média e o desvio padrão também são utilizados neste estudo como base para a comparação estatística entre os algoritmos.

Aplicando o teste de Friedman, com nível de significância de 0,05, para os três algoritmos meta-heurísticos (GRASP, SA e ILS), considerando todas as instâncias, obtém-se um *p-valor* de $2,23 \times 10^{-5}$, indicando que há uma diferença estatística significativa entre as três implementações. Para verificar quais pares são diferentes entre si, foi aplicado o teste *post-hoc* de Nemenyi. Esse teste retornou os *p-valores* conforme a Tabela 6, indicando que apenas os algoritmos GRASP e ILS são diferentes estatisticamente entre si.

Tabela 6 – Teste *post-hoc* entre as meta-heurísticas GRASP, SA, ILS para todas as instâncias

Algoritmo 1	Algoritmo 2	*p-valor*
GRASP	SA	0,9573
GRASP	ILS	0,0307
SA	ILS	0,0633

Dependendo do tamanho da instância, os três algoritmos produzem resultados melhores ou piores em relação aos outros. Portanto, para uma análise estatística mais detalhada, os mesmos testes foram aplicados aos três algoritmos em cada tamanho de instância. Esses resultados estão apresentados na Tabela 7. A coluna "Friedman" *p-valor* apresenta o resultado obtido pela aplicação do teste de Friedman com nível de significância de 0,05. Apenas para o tamanho GG os três algoritmos meta-heurísticos não possuem diferença estatística em seus resultados. Por não possuir diferença, não haveria necessidade de um teste *post-hoc*. Além disso, as instâncias GG apresentam uma amostra pequena (três instâncias) para que o teste seja confiável, sendo recomendada pelo menos uma amostra maior do que cinco. Para os demais tamanhos de instâncias, o teste de Friedman indica diferença estatística entre os algoritmos, por isso há necessidade do teste *post-hoc* Nemenyi, que indica entre quais pares há diferença.

Tabela 7 – Teste *post-hoc* entre os algoritmos meta-heurísticos GRASP, SA e ILS por grupos de instâncias

	Friedman	Teste Nemenyi		
	p-valor	Algoritmo 1	Algoritmo 2	*p-valor*
P	2×10^{-8}	GRASP	SA	0,3712
		GRASP	ILS	0,2653
		SA	ILS	0,0105
M	0,025961	GRASP	SA	0,6298
		GRASP	ILS	0,0935
		SA	ILS	0,4733
G	0,029508	GRASP	SA	0,1590
		GRASP	ILS	0,0416
		SA	ILS	0,8291
GG	0,716531	NA		

Na Tabela 7, coluna "Teste Nemenyi", os *p-valores* obtidos apontam a existência de diferença estatística entre SA e ILS para o tamanho P e entre GRASP e ILS para o tamanho G de instâncias. Para o tamanho M, o teste de Nemenyi não apontou diferença entre nenhum par. Nesse caso, a comparação é feita pelos valores de média dentre os pares. Para o tamanho M, o GRASP obteve média de 2024,8, SA média de 2020,6 e ILS média de 2018,8.

Como GRASP e ILS estão mais distantes entre si, esses são os dois algoritmos diferentes estatisticamente.

Ao iniciar a comparação dos algoritmos meta-heurísticos desenvolvidos com os da literatura (Aquino *et al*., 2019), foram observadas inconsistências nos resultados com a alocação de ordens não alocáveis e, assim, resultados inviáveis. Dessa forma, algumas instâncias foram retiradas da comparação.

Para realizar a comparação com a literatura, foi escolhido o melhor algoritmo dentre os três implementados neste trabalho. O algoritmo escolhido foi o ILS, que obteve melhores resultados de forma geral, conforme evidenciado na Tabela 5. No trabalho de Aquino et al. (2019), algoritmos diferentes também obtiveram diferentes desempenhos dependendo do tamanho da instância. Para fins de comparação, foi escolhido o BRKMA, porque é o algoritmo que obteve melhores resultados nas instâncias maiores e na real. A Tabela 8 apresenta o comparativo entre os estudos.

Tabela 8 – Comparativo entre ILS e literatura por resultados em grupos de instâncias

Resultados	**Todas as instâncias**	**P**	**M**	**G**	**GG**
Piores	10,57%	4,44%	9,09%	20,00%	0,00%
Melhores	14,63%	0,00%	18,18%	30,00%	100,00%
Iguais	74,80%	95,56%	72,73%	50,00%	0,00%

Conforme apresentado na Tabela 8, no conjunto de todas as instâncias, o ILS apresenta resultados melhores em 14,63% das instâncias, piores em 10,57% e iguais em 74,8%. Para as instâncias pequenas, os resultados são 4,44% piores, o restante apresenta resultados iguais. Porém, para instâncias M, G e GG, o número de instâncias com melhores resultados ultrapassa os piores, chegando a 100% de resultados melhores para as instâncias GG. Em relação à instância real, o resultado obtido pelo GRASP foi 40,5% melhor que o apresentado por Aquino et al. (2019).

Os resultados completos obtidos pelos algoritmos desenvolvidos em todas as instâncias e execuções estão disponibilizados em https://docs.google.com/spreadsheets/d/1_iajAPkJbFNVgEQie_jAWGJa80DxXimvTwpEJX3q-wY/edit?usp=sharing

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foram implementados algoritmos heurísticos construtivos e algoritmos meta-heurísticos para gerar soluções competitivas para o PPOMPLP, visando à redução de manutenções não realizadas e de custo com equipes. Os resultados obtidos pelos três métodos meta-heurísticos foram comparados entre si e com a literatura disponível. Dentre os algoritmos implementados neste trabalho, o ILS é o que apresentou melhor repetibilidade dos resultados e soluções melhores de forma geral, obtendo o melhor resultado em 99% das instâncias tamanho P e em 71% das instâncias tamanho M, G e GG. Apesar disso, o melhor resultado obtido para a instância real foi alcançado pelo GRASP. Estatisticamente, apenas GRASP e ILS são diferentes entre si, quando os algoritmos são analisados em relação a todas as instâncias.

Na comparação dos resultados obtidos pelo ILS e o BRKMA de Aquino et al. (2019), os resultados do ILS foram melhores em 14,63% das instâncias, iguais para 74,8% das instâncias e piores para 10,57%, quando comparados em relação a todas as instâncias. Quando a comparação é feita de acordo com o tamanho das instâncias, o ILS apresenta pior desempenho nas instâncias tamanho P e melhor desempenho para as instâncias tamanho M, G e GG. Para a instância real, o valor obtido pelo ILS é 40,5% melhor que o valor obtido pelo BRKMA.

A comparação dos algoritmos desenvolvidos neste trabalho com a literatura tem como limitação a utilização de linguagens de programação diferentes, o processamento em *threads* paralelas, realizado por Aquino et al. (2019) e não neste trabalho, e a utilização de máquinas diferentes para o processamento.

Para trabalhos futuros, sugere-se considerar a programação das ordens de manutenção não atendidas, inserindo-as na programação com atraso, em vez de simplesmente não as alocar. Isso geraria um novo problema com um objetivo de otimização diferente.

REFERÊNCIAS

ALFARES, H., MOHAMMED, A., E GHALEB, M. (2021). Two-machine scheduling with aging effects and variable maintenance activities. **Computers & Industrial Engineering,** 160:107586.

ALFARES, H. K. (2022). Plant shutdown maintenance workforce team assignment and job scheduling. **Journal of Scheduling**, 25(3):321–338.

ALIDAEE, B. E ROSA, D. (1997). Scheduling parallel machines to minimize total weighted and unweighted tardiness. **Computers & Operations Research,** 24(8):775–788.

AQUINO, R. D., CHAGAS, J. B. C., E SOUZA, M. J. F. (2019). Abordagem exata e heurísticas para o problema de planejamento de ordens de manutenção de longo prazo: Um estudo de caso industrial de larga escala. **Pesquisa Operacional para o Desenvolvimento**, 11(3):159–182.

AVALOS-ROSALES, O *et al.* (2018). Including preventive maintenance activities in an unrelated parallel machine environment with dependent setup times. **Computers & Industrial Engineering,** 123:364–377.

BROWN, I. E MUES, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. **Expert Systems with Applications**, 39(3):3446–3453.

DELORME, M., IORI, M., E MENDES, N. F. (2021). Solution methods for scheduling problems with sequence-dependent deterioration and maintenance events. **European Journal of Operational Research,** 295(3):823–837.

EBRAHIMIPOUR, V., NAJJARBASHI, A., E SHEIKHALISHAHI, M. (2015). Multi-objective modeling for preventive maintenance scheduling in a multiple production line. **Journal of Intelligent Manufacturing, 26(1):111–122.**

ERTEM, M., AS' AD, R., AWAD, M., E AL-BAR, A. (2022). Workers-constrained shutdown maintenance scheduling with skills flexibility: Models and solution algorithms. **Computers & Industrial Engineering,** p. 108575.

FAKHER, H. B., NOURELFATH, M., E GENDREAU, M. (2016). A cost minimisation model for joint production and maintenance planning under quality constraints. **International Journal of Production Research,** 55(8):2163–2176.

FEO, T. A. E RESENDE, M. G. (1995). Greedy randomized adaptive search procedures. **Journal of Global Optimization**, 6(2):109–133.

FRIEDMAN, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. **The Annals of Mathematical Statistics**, 11(1):86–92.

FU, X., *et al.* (2019). A three-level particle swarm optimization with variable neighbourhood search algorithm for the production scheduling problem with mould maintenance. **Swarm and Evolutionary Computation**, 50:100572.

HEDJAZI, D. (2015). Scheduling a maintenance activity under skills constraints to minimize total weighted tardiness and late tasks. **International Journal of Industrial Engineering Computations,** 6(2):135–144.

KIRKPATRICK, S., GELATT, C. D., E VECCHI, M. P. (1983). Optimization by simulated annealing. **Science,** 220(4598):671–680.

LEE, H. E CHA, J. H. (2016). New stochastic models for preventive maintenance and maintenance optimization. **European Journal of Operational Research**, 255(1):80–90.

LEVITIN, G., XING, L., E DAI, Y. (2021). Optimal operation and maintenance scheduling in m-out-of-n standby systems with reusable elements. **Reliability Engineering & System Safety**, 211:107582.

LIU, Y *et al.* (2021). Integrated production planning and preventive maintenance scheduling for synchronized parallel machines. **Reliability Engineering & System Safety**, 215:107869.

LÓPEZ-IBÁÑEZ, M *et al.* (2016). The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, 3:43–58.

LOURENÇO, H. R., MARTIN, O. C., E STÜTZLE, T. (2019). Iterated local search: Framework and applications. In **Handbook of metaheuristics**, p. 129–168. Springer.

LU, S *et al.* (2021). A hybrid DBH-VNS for high-end equipment production scheduling with machine failures and preventive maintenance activities. **Journal of Computational and Applied Mathematics**, 384:113195.

MA, J *et al.* (2022). A comprehensive comparison among metaheuristics (MHs) for geohazard modeling using machine learning: Insights from a case study of landslide displacement prediction. **Engineering Applications of Artificial Intelligence,** 114:105150.

MARMIER, F., VARNIER, C., E ZERHOUNI, N. (2009). Static et dynamic scheduling of maintenance activities under the constraints of skills. **Journal of Operations and Logistics**, 2(3):I–1.

PINEDO, M. (2016). **Scheduling**: theory, algorithms, and systems. Springer, Cham, 5 edition.

QI, X., CHEN, T., E TU, F. (1999). Scheduling the maintenance on a single machine. **Journal of the Operational Research Society,** 50(10):1071–1078.

RUIZ-TORRES, A. J., PALETTA, G., E M'HALLAH, R. (2016). Makespan minimisation with sequence dependent machine deterioration and maintenance events. *International* **Journal of Production Research**, 55(2):462–479.

SHELDON, M. R., FILLYAW, M. J., E THOMPSON, W. D. (1996). The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. **Physiotherapy Research International,** 1(4):221–228.

UPASANI, K *et al.* (2017). Distributed maintenance planning in manufacturing industries**. Computers & Industrial Engineering, 108:1–14.**

VALLADA, E. E RUIZ, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. **European Journal of Operational Research**, 211 (3):612–622.

WANG, J. E MIAO, Y. (2021). Optimal preventive maintenance policy of the balanced system under the semi-markov model. **Reliability Engineering & System Safety**, 213:107690.

WANG, Y *et al.* (2018). Source term estimation of hazardous material releases using hybrid genetic algorithm with composite cost functions. **Engineering Applications of Artificial Intelligence**, 75:102–113.

XU, J *et al.* (2019). An iterated local search and tabu search for two-parallel machine scheduling problem to minimize the maximum total completion time. **Journal of Information and Optimization Sciences**, 40(3):751–766.

Como Referenciar este Artigo, conforme ABNT:

SANTOS, A. A; MARTINS, A. X; SOUSA, M. J. F; MACHADO, R. H. C. Métodos Heurísticos e Meta-Heurísticos para a Resolução do Problema de Sequenciamento de Ordens de Manutenção Preventiva de Longo Prazo. **Rev. FSA**, Teresina, v. 20, n. 10, art. 8, p. 175-202, out. 2023.

Contribuição dos Autores	**A. A. Santos**	**A. X. Martins**	**M. J. F. Sousa**	**R. H. C. Machado**
1) concepção e planejamento.	X	X	X	
2) análise e interpretação dos dados.	X	X	X	
3) elaboração do rascunho ou na revisão crítica do conteúdo.	X	X	X	X
4) participação na aprovação da versão final do manuscrito.	X	X	X	X